STATUS OF THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Original) A method comprising:

allocating a first node;

storing a value in a thread-local variable field in the first node; and

identifying a second node in a data structure allocated by a runtime environment while an operating system associated with the runtime environment is in an unlocked condition.

2. (Original) A method as defined in claim 1, further comprising:

storing a second value in a stack address field in the first node, wherein the stack address field is associated with a stack allocated by the operating system; and

establishing a relationship between the first and second nodes in the data structure based on a value of the stack address field.

3. (Original) A method as defined in claim 2, wherein the relationship between the first and second nodes comprises a value in a stack address field in the second node that is greater than the second value in the stack address field in the first node.

4. (Original) A method as defined in claim 1, wherein the thread-local variable field comprises a high-level language data structure.

5. (Original) A method as defined in claim 4, wherein the high-level language data structure comprises at least one of a C/C++ structure, a C++ class, a Java class, and a C# class.

6. (Original) A method as defined in claim 1, wherein the thread-local variable field comprises an indirect reference.

7. (Original) A method as defined in claim 6, wherein the indirect reference comprises at least one of a C/C++ pointer, a Java reference, a C++ reference, a C# reference, and an assembly language indirect memory reference.

8. (Original) A method as defined in claim 1, wherein the first node comprises at least one of a statically allocated node and a dynamically allocated node.

9. (Original) A method as defined in claim 1, wherein the data structure comprises at least one of a linked list-based data structure, an array, a queue-based data structure, a stack-based data structure, and a tree-based data structure.

10. (Original) A method as defined in claim 1, wherein the runtime environment comprises a virtual machine.

11. (Original) An apparatus comprising:

a memory; and

a processor coupled to the memory and configured to:

allocate a first node;

store a value in a thread-local variable field in the first node; and

identify a second node in a data structure allocated by a runtime environment while an operating system associated with the runtime environment is in an unlocked condition.

12. (Original) An apparatus as defined in claim 11, wherein the processor is further configured to:

store a second value in a stack address field in the first node, wherein the stack address field is associated with a stack allocated by the operating system; and

establish a relationship between the first and second nodes in the data structure based on a value of the stack address field.

13. (Original) An apparatus as defined in claim 12, wherein the relationship between the first and second nodes comprises a value in a stack address field in the second node that is greater than the second value in the stack address field in the first node.

14. (Original) An apparatus as defined in claim 11, wherein the thread-local variable field comprises a high-level language data structure.

15. (Original) An apparatus as defined in claim 14, wherein the high-level language data structure comprises at least one of a C/C++ structure, a C++ class, a Java class, and a C# class.

16. (Original) An apparatus as defined in claim 11, wherein the thread-local variable field comprises an indirect reference.

17. (Original) An apparatus as defined in claim 16, wherein the indirect reference comprises at least one of a C/C++ pointer, a Java reference, a C++ reference, a C# reference, and an assembly language indirect memory reference.

18. (Original) An apparatus as defined in claim 11, wherein the first node comprises at least one of a statically allocated node and a dynamically allocated node.

19. (Original) An apparatus as defined in claim 11, wherein the data structure comprises at least one of a linked list-based data structure, an array, a queue-based data structure, a stack-based data structure, and a tree-based data structure.

20. (Original) An apparatus as defined in claim 11, wherein the runtime environment comprises a virtual machine.

21. (Original) A machine readable medium having instructions stored thereon that, when executed, cause a machine to:

allocate a first node;

store a value in a thread-local variable field in the first node; and

identify a second node in a data structure allocated by a runtime environment while an operating system associated with the runtime environment is in an unlocked condition.

22. (Original) A machine readable medium as defined in claim 21, having instructions stored thereon that, when executed, cause the machine to:

store a second value in a stack address field in the first node, wherein the stack address field is associated with a stack allocated by the operating system; and

establish a relationship between the first and second nodes in the data structure based on a value of the stack address field.

23. (Original) A machine readable medium as defined in claim 22, wherein the relationship between the first and second nodes comprises a value in a stack address field in the second node that is greater than the second value in the stack address field in the first node.

24. (Original) A machine readable medium as defined in claim 21, wherein the thread-local variable field comprises a high-level language data structure.

25. (Original) A machine readable medium as defined in claim 24, wherein the high-level language data structure comprises at least one of a C/C++ structure, a C++ class, a Java class, and a C# class.

26. (Original) A machine readable medium as defined in claim 21, wherein the thread-local variable field comprises an indirect reference.

27. (Original) A machine readable medium as defined in claim 26, wherein the indirect reference comprises at least one of a C/C++ pointer, a Java reference, a C++ reference, a C# reference, and an assembly language indirect memory reference.

28. (Original) A machine readable medium as defined in claim 21, wherein the first node comprises at least one of a statically allocated node and a dynamically allocated node.

29. (Original) A machine readable medium as defined in claim 21, wherein the data structure comprises at least one of a linked list-based data structure, an array, a queue-based data structure, a stack-based data structure, and a tree-based data structure.

30. (Original) A machine readable medium as defined in claim 21, wherein the runtime environment comprises a virtual machine.